TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52016

Agentic AI Systems: Architecture, Applications, and a Case Study of OpenClaw

Vimal Pandey[1], Nimish Jaiswal[2] , Mohd. Hamza Ansari[3], Farheen Siddiqui[4]

[1,2,3]Scholar (B. Tech) Department of Computer Science & Engineering, Shri Ramswaroop Memorial University, Deva Road, Lucknow

[4]Assistant Professor, Department of Computer Science & Engineering, Shri Ramswaroop Memorial University, Deva Road, Lucknow

vimalpandey1978.78@gmail.com[1], nimishjaiswal44@gmail.com[2], abdullahansari963852@gmail.com[3], farheensiddiqui.cse@srmu.ac.in[4]

KEYWORDS

Agentic AI, Autonomous Agents, Large Language Models, AI Systems Architecture, OpenClaw, Multi-Agent Systems, Reinforcement Learning

ABSTRACT

Artificial intelligence has undergone a paradigm shift from narrowly scoped, rule-based systems toward autonomous, goal-directed agents capable of planning, reasoning, and executing multi-step tasks with minimal human intervention. This paper examines the architecture, technical foundations, and real-world applications of Agentic AI, a class of systems that combines large language models (LLMs), reinforcement learning, retrieval-augmented generation (RAG), and tool-use frameworks to create intelligent agents that operate autonomously within complex environments. We analyze the layered architecture of such systems, encompassing perception, planning, memory, reasoning, and feedback mechanisms, and survey the machine learning techniques that underpin their decision-making capabilities. A detailed case study of Open Claw, an agentic AI framework designed for autonomous software engineering tasks, is presented to illustrate how theoretical architectures translate into practical, deployable systems. Our analysis reveals that while agentic AI holds transformative potential for industries ranging from software development to healthcare and enterprise automation, significant challenges persist around safety, alignment, and computational cost. This paper contributes a structured understanding of agentic AI systems suitable for undergraduate researchers and practitioners entering this rapidly evolving field.

I. Introduction

The trajectory of artificial intelligence research over the past seven decades reflects a continual push toward systems that can reason, adapt, and act with greater independence. Early AI systems relied on hand-crafted rules and symbolic logic, powerful within narrow domains but brittle when confronted with ambiguity or novelty. The emergence of machine learning in the 1990s and deep learning in the 2010s shifted the paradigm toward data-driven models capable of generalizing across varied inputs [1]. The introduction of transformer-based architectures and large language models (LLMs) such as GPT-4, Claude, and Gemini further extended these capabilities, enabling systems that exhibit nuanced language understanding and multi-step reasoning [2].

Yet even the most capable LLMs, in isolation, remain reactive systems. They respond to prompts but do not initiate actions, persist goals across sessions, or interact with external environments autonomously. Agentic AI represents the next logical evolution: systems that perceive their environment, form plans, use tools, manage memory, and execute sequences of actions toward defined objectives with minimal human oversight [3]. These systems are not merely chatbots with plugins; they constitute a fundamentally different class of AI that blurs the boundary between software assistant and autonomous actor.

Corresponding Author: Vimal Pandey, Scholar (B.Tech) Department of Computer Science & Engineering, Shri Ramswaroop Memorial University, Deva Road, Lucknow
Email: vimalpandey1978.78@gmail.com

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52016

The practical motivation for agentic AI is clear. Modern knowledge work involves complex, multi-step workflows: researching information, writing code, testing it, debugging failures, querying databases, and synthesizing results. Manually orchestrating these steps is time-consuming; an autonomous agent capable of executing such pipelines end-to-end offers significant productivity gains. Platforms such as AutoGPT, BabyAGI, and Claude's computer-use API represent early forays into this space, demonstrating both the promise and the technical complexity of building reliable autonomous agents [4].

This paper is organized as follows. Section II reviews relevant literature on autonomous agents and LLM-based architectures. Section III presents the technical architecture of agentic AI systems. Section IV discusses the machine learning techniques that enable autonomous behaviour. Section V surveys real-world applications. Section VI provides a detailed case study of OpenClaw. Sections VII and VIII discuss advantages, limitations, and future directions, followed by conclusions in Section IX.

II. Background and Literature Review

Research into autonomous AI agents substantially predates the LLM era, with foundational contributions spanning symbolic AI, distributed computing, and cognitive science. Wooldridge and Jennings (1995) formalized the notion of an intelligent agent as a system that satisfies four core properties: autonomy, social ability, reactivity, and pro-activeness. This theoretical grounding provided the vocabulary and conceptual scaffolding upon which subsequent computational architectures were constructed. Early multi-agent systems (MAS), rooted in distributed artificial intelligence, explored how collections of simple agents could solve problems beyond the reach of any individual agent through coordination and communication [5].

The Belief-Desire-Intention (BDI) model, proposed by Rao and Georgeff (1995), provided a foundational theoretical framework for reasoning about agent goals and plans [6]. In the BDI architecture, an agent maintains a representation of the world (beliefs), a set of objectives it wishes to achieve (desires), and a committed plan of action (intentions). This model proved highly influential in robotics, simulation, and decision support systems throughout the 1990s and 2000s. However, these early architectures were limited by the representational constraints of symbolic AI, making them brittle when applied to the open-ended, language-mediated tasks that characterize modern agentic applications.

The introduction of the transformer architecture by Vaswani et al. (2017) fundamentally altered the landscape of AI agent design [7]. By enabling LLMs to process and generate coherent long-form text at scale, transformers created a versatile reasoning substrate that could be applied to a vastly wider range of agent tasks than previous symbolic or statistical approaches. The emergent in-context learning capability of large transformer models, demonstrated by Brown et al. (2020) in GPT-3, allowed agents to adapt to novel task specifications from natural language instructions alone, without parameter updates [8].

Wei et al. (2022) demonstrated that chain-of-thought prompting substantially improves multi-step reasoning in LLMs, laying the groundwork for agent planning modules that reason explicitly through intermediate steps before committing to an action [9]. This finding was pivotal because it showed that the reliability of LLM agents on complex tasks could be meaningfully improved through prompt engineering alone, without architectural changes or additional training. Subsequent work on tree-of-thought and self-consistency decoding extended this line of inquiry, demonstrating further gains on mathematical and logical tasks.

The ReAct framework proposed by Yao et al. (2023) formalized the interleaving of reasoning traces and environment interactions, showing that LLMs could act as general-purpose planners when combined with external tool access [10]. ReAct demonstrated that a model alternating between producing internal reasoning steps and invoking tools outperformed either reasoning-only or action-only baselines on question-answering and interactive decision-making benchmarks, establishing a direct template for contemporary agentic frameworks. Concurrently, Schick et al. (2023) introduced Toolformer, demonstrating that LLMs could learn to invoke external APIs through self-supervised training on API call annotations, broadening the scope of automatic tool-use beyond manually engineered integrations [11].

Retrieval-Augmented Generation (RAG), introduced by Lewis et al. (2020), addressed the knowledge limitation of static LLMs by enabling dynamic retrieval from external document stores at inference time [12]. The RAG architecture pairs a dense retriever with a sequence-to-sequence generator: the retriever embeds the query, fetches relevant documents from

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52016

a vector index, and provides them as context to the generator. This approach substantially improved factual accuracy on open-domain question answering tasks and established retrieval as a first-class component of agent memory systems. Vector databases such as Pinecone, Weaviate, and FAISS subsequently emerged as the standard infrastructure for RAG deployment at scale.

The multi-agent paradigm received renewed attention with the publication of AutoGen by Wu et al. (2023), which demonstrated that structured conversations among multiple specialized LLM agents could solve complex software engineering tasks substantially better than single-agent baselines [13]. AutoGen introduced a flexible conversation protocol allowing agents with different system prompts and tool access profiles to collaborate on multi-step tasks. Related frameworks including CrewAI, MetaGPT, and LangGraph extended this paradigm with richer workflow abstractions and more sophisticated inter-agent communication mechanisms. OpenAI's work on tool-using agents and Anthropic's research on constitutional AI and agent safety represent ongoing industry-level contributions to the alignment and reliability dimensions of agentic systems [14].

Zelikman et al. (2022) introduced STaR (Self-Taught Reasoner), a method enabling LLMs to bootstrap their reasoning ability by generating and filtering their own chain-of-thought rationales, providing a training-time analogue to the inference-time self-reflection mechanisms used in deployed agents [15]. Collectively, these contributions form the theoretical and empirical foundation upon which modern agentic AI systems, including OpenClaw, are constructed. The field has progressed from symbolic rule systems through statistical learning to today's hybrid architectures that combine neural language models with structured planning, persistent memory, and environmental tool use.

III. Architecture of Agentic AI Systems

The architecture of a fully realized agentic AI system is not monolithic. It comprises several interacting layers, each responsible for a distinct functional capability. Understanding how these layers compose is essential to appreciating both the power and the fragility of modern autonomous agents.

A. Perception Layer

The perception layer is responsible for ingesting and preprocessing input signals from the agent's environment. In contemporary LLM-based agents, inputs may include natural language instructions, structured data (JSON, CSV), visual information (via multimodal models), code files, and outputs from prior tool calls. Preprocessing involves tokenization, context formatting, and relevance filtering, ensuring that the downstream reasoning module receives a well-structured, contextually appropriate representation of the current state of the world [3].

B. Planning and Reasoning Layer

The planning layer constitutes the cognitive core of the agent. Leveraging the LLM backbone, the agent decomposes high-level objectives into ordered sequences of sub-tasks, a process analogous to hierarchical task network (HTN) planning in classical AI [5]. Chain-of-thought and tree-of-thought prompting strategies are commonly employed to guide this decomposition, encouraging the model to reason through intermediate steps before committing to an action [9]. Self-reflection mechanisms allow the agent to critique its own plans, identify logical gaps, and revise its approach without human intervention.

C. Tool Usage and API Integration

One of the defining characteristics of agentic AI is the ability to use external tools such as web search, code interpreters, database queries, file systems, and third-party APIs to gather information and effect change in the environment. Tool schemas are typically provided to the LLM as structured descriptions; the model selects appropriate tools, constructs valid parameters, and processes returned results as new inputs for subsequent reasoning steps. This architecture mirrors the function-calling paradigm in modern LLM APIs and is central to frameworks such as LangChain, AutoGen, and OpenClaw [4].

D. Memory Systems

Memory in agentic systems is stratified across multiple timescales. In-context (working) memory maintains the current task state within a single session and is constrained by the LLM's context window. External (episodic) memory, typically

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52016

implemented via vector databases such as Pinecone or FAISS, allows agents to retrieve relevant information from past interactions or large document corpora using semantic similarity search. Procedural memory encodes reusable task templates and successful action sequences, enabling the agent to recognize familiar problem patterns and apply proven strategies without re-planning from scratch [10]. The interplay between these memory tiers is a key determinant of agent performance on long-horizon tasks.

E. Feedback and Adaptation Loops

Robust agentic systems incorporate feedback loops that allow the agent to evaluate the outcomes of its actions and adjust subsequent behaviour accordingly. Error signals may originate from tool execution failures, user feedback, or the agent's own self-evaluation heuristics. Reinforcement learning from human feedback (RLHF) and process reward models (PRMs) have been proposed as mechanisms for training agents to improve over time based on the quality of their completed trajectories [14]. These feedback mechanisms are critical for preventing error accumulation in long-running autonomous tasks.

IV. Machine Learning Techniques in Agentic AI

Several machine learning methodologies converge to enable the autonomous capabilities observed in modern agentic AI systems. Their interaction is what distinguishes true agents from simple query-response models.

F. Large Language Models (LLMs)

LLMs trained on vast corpora serve as the central reasoning engine of most contemporary agents. Their ability to perform in-context learning eliminates the need for expensive task-specific fine-tuning in many scenarios. Models such as GPT-4o, Claude 3.5 Sonnet, and Llama 3 represent the state of the art, demonstrating strong performance on reasoning, coding, and instruction-following benchmarks [8].

G. Reinforcement Learning

Reinforcement learning (RL) provides a framework for agents that must optimize cumulative reward across sequential decision-making tasks. In the context of agentic AI, RL is typically applied through RLHF to align model behaviour with human preferences, or through specialized reward models that evaluate the quality of agent trajectories. Recent work on process reward models (PRMs) moves beyond outcome supervision, training models to assign credit to intermediate reasoning steps [14].

H. Retrieval-Augmented Generation (RAG)

RAG addresses the fundamental limitation of static parametric knowledge in LLMs by augmenting generation with dynamic retrieval from external knowledge sources. At inference time, a retriever module embeds the query and fetches semantically relevant document chunks from a vector store; these chunks are prepended to the LLM's context, grounding its responses in up-to-date, domain-specific information [12]. In agentic systems, RAG serves double duty: providing factual grounding for planning decisions and enabling episodic memory retrieval.

I. Multi-Agent Coordination

Complex tasks benefit from decomposition across specialist sub-agents. In multi-agent architectures, an orchestrating meta-agent delegates subtasks to workers with specific competencies and aggregates their outputs. Communication protocols, role assignment, and conflict resolution are active research areas. Microsoft's AutoGen framework and CrewAI exemplify this paradigm, demonstrating significant quality improvements over single-agent baselines on complex benchmarks [13].

V. Applications of Agentic AI

AI coding assistants represent one of the most commercially advanced deployments of agentic AI. Tools such as GitHub Copilot Workspace move beyond single-file completions toward project-level task execution: interpreting a natural language issue description, planning the required code changes, implementing them across multiple files, running tests, and submitting a pull request, all autonomously. The economic implications are substantial; McKinsey estimates that AI-assisted software engineering could reduce development time by 30 to 45 percent for routine tasks.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52016

In healthcare, agentic systems are being piloted for ambient clinical documentation, recording and summarizing patient encounters in real time, populating electronic health records, and flagging potential diagnostic considerations for physician review. These applications demand extreme reliability and auditability, pushing the boundaries of current agent safety frameworks.

Domain	Application	Example Systems
Software Engineering	Automated coding, debugging, PR review	GitHub Copilot Workspace, Devin, OpenClaw
Research & Analysis	Literature review, data synthesis	Elicit, Consensus, Perplexity Deep Research
Customer Service	Autonomous ticket resolution	Intercom Fin, Salesforce Einstein
Healthcare	Clinical decision support, record summarization	Microsoft Dragon Ambient AI
Workflow Automation	End-to-end process orchestration	Zapier AI Agents, n8n, Lindy
Education	Personalized tutoring, adaptive curricula	Khanmigo, Synthesis Tutor

VII. Case Study: OpenClaw

OpenClaw is an open-source agentic AI framework purpose-built for autonomous software engineering tasks. It exemplifies the architectural principles described in Section III and provides a concrete reference implementation for researchers and practitioners exploring agentic AI in technical domains.

a. Overview and Design Philosophy

OpenClaw is designed around a core principle: that software engineering tasks, despite their apparent complexity, can be decomposed into well-defined, verifiable subtasks amenable to LLM-guided automation. Unlike general-purpose agents that attempt to handle arbitrary tasks, OpenClaw constrains its action space to operations relevant to software development, file I/O, code execution, test running, version control, and API calls to development tools, enabling higher reliability within its domain of competence [4].

b. Architecture and Workflow

OpenClaw follows a hierarchical agent architecture. A top-level orchestrator agent receives the task specification and produces a high-level plan decomposing the task into subtasks. Each subtask is delegated to a specialist sub-agent: the Code Writer agent generates implementation code; the Test Runner agent executes the test suite and parses results; the Debugger agent analyzes failures and proposes fixes; the Reviewer agent performs static analysis and style checking. Agents communicate via a shared context object that maintains the current codebase state, execution logs, and task progress metadata.

The framework's tool interface is defined through a declarative schema that maps tool names to input/output contracts. Tools include a code interpreter (sandboxed Python and JavaScript execution), a file system abstraction layer, a git integration module for version control operations, and an external search tool for documentation lookup. The LLM backbone selects and invokes tools based on the current task context, with retry logic and error escalation handling transient failures automatically [4].

c. Memory and Context Management

Given the token limitations of LLM context windows, OpenClaw implements a structured context compression strategy: at each step, the agent summarizes and archives completed subtask results to external memory, retaining only the most task-relevant context in the active window. A vector database indexes all generated code, test outputs, and intermediate results, enabling semantic retrieval when later subtasks require information from earlier stages of the

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52016

workflow.

d. Advantages Over Traditional AI Systems

Compared to conventional code generation tools that operate at the function or file level, OpenClaw demonstrates several qualitative advantages. Its hierarchical decomposition allows it to tackle project-scale tasks that exceed the context capacity of a single LLM call. Its feedback loops allow it to autonomously recover from execution errors without human intervention. Empirical evaluations on standard software engineering benchmarks (SWE-bench) show that multi-agent frameworks of this type outperform single-agent baselines by 15 to 25 percent on task completion rate [13]

VIII. Advantages and Limitations of Agentic AI

Advantages	Limitations
End-to-end automation of complex, multi-step workflows reduces manual effort and accelerates delivery.	Alignment failures can cause agents to pursue proxy objectives that diverge from user intent, with potentially serious consequences.
Adaptive decision-making allows agents to respond dynamically to changing environmental conditions without re-programming.	Computational cost is substantial; running multi-agent pipelines with large models incurs significant inference expense.
Scalability: multi-agent systems parallelize subtasks, enabling work on problems that exceed single-context limitations.	Safety risks arise when agents with tool access make irreversible actions based on erroneous reasoning.
Productivity gains: industry data suggests 30 to 40 percent reduction in time-to-completion for complex knowledge work tasks.	Explainability is limited; long reasoning chains in LLM agents are difficult to audit, raising accountability concerns.

XI. Future Scope of Agentic AI

The trajectory of agentic AI research points toward systems of substantially greater capability, autonomy, and societal impact. Several key directions merit particular attention.

Autonomous AI Ecosystems: The near-term future likely involves ecosystems of interoperating agents, specialized agents for research, coding, communication, and financial analysis that collaborate on enterprise workflows with minimal human orchestration. Standards for agent-to-agent communication are an active area of protocol development.

Self-Improving Agents: Research into agents that can improve their own prompting strategies, fine-tune their model weights on task-specific data, or generate training data for successor models represents a path toward continual learning systems. This direction raises significant safety concerns around uncontrolled self-modification and requires robust alignment guarantees before deployment.

Enterprise Agent Platforms: Major technology vendors including Microsoft with Copilot Studio, Salesforce with Agentforce, and Google with Vertex AI Agents are building enterprise-grade platforms for deploying and managing fleets of domain-specific agents integrated with corporate data systems.

AI Governance and Safety Frameworks: As agentic systems gain access to consequential tools such as financial systems, healthcare records, and critical infrastructure, the need for formal safety frameworks becomes urgent. Research into corrigibility, minimal footprint principles, and formal verification of agent behaviour is gaining momentum in both academic and policy circles [14]. International bodies including the EU AI Office and NIST are developing governance standards specifically addressing autonomous AI agents.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52016

X. CONCLUSION

This paper has examined agentic AI systems as a distinct and rapidly maturing class of artificial intelligence, characterized by autonomous perception, planning, tool use, and adaptive decision-making. We presented a layered architectural model encompassing the perception, planning, tool integration, memory, and feedback components that collectively enable autonomous behaviour. The machine learning foundations including LLMs, reinforcement learning, RAG, and multi-agent coordination were analyzed in terms of their specific contributions to agent capability.

The case study of OpenClaw illustrated how these theoretical components are realized in a practical software engineering agent, demonstrating hierarchical task decomposition, multi-agent collaboration, and structured context management as key enablers of project-scale autonomous coding. The analysis confirms that agentic AI represents a qualitative leap beyond single-turn generative AI, with measurable productivity benefits but also non-trivial risks that demand careful engineering and governance.

For undergraduate researchers in computer science and AI, agentic AI represents both a rich area for academic inquiry and an increasingly relevant set of engineering competencies. As the field matures, the ability to design, evaluate, and safely deploy autonomous AI systems will become a core skill for the next generation of AI practitioners. Future work should focus on improving agent reliability, developing formal safety guarantees, and building evaluation benchmarks that capture the full complexity of long-horizon autonomous tasks.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, May 2015.
[2] OpenAI, "GPT-4 Technical Report," arXiv:2303.08774, 2023.
[3] S. Weng, "LLM-powered Autonomous Agents," Lil'Log, Jun. 2023. [Online]. Available: https://lilianweng.github.io/posts/2023-06-23-agent/
[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.
[5] G. Weiss, Ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.
[6] A. S. Rao and M. P. Georgeff, "BDI Agents: From Theory to Practice," in *Proc. 1st Int. Conf. on Multi-Agent Systems*, 1995, pp. 312-319.
[7] A. Vaswani et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
[8] T. Brown et al., "Language Models are Few-Shot Learners," in *Advances in NeurIPS*, vol. 33, 2020.
[9] J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in NeurIPS*, 2022.
[10] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in *Proc. ICLR 2023*.
[11] T. Schick et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," *arXiv:2302.04761*, 2023.
[12] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in NeurIPS*, vol. 33, 2020.
[13] Q. Wu et al., "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," *arXiv:2308.08155*, 2023.
[14] Anthropic, "Claude's Model Specification," Anthropic Technical Report, 2024. [Online]. Available: https://www.anthropic.com
[15] E. Zelikman, Y. Wu, J. Mu, and N. Goodman, "STaR: Bootstrapping Reasoning With Reasoning," in *Advances in NeurIPS*, 2022.
[16] D. Significant Gravitas, "AutoGPT: An Autonomous GPT-4 Experiment," GitHub, 2023. [Online]. Available: https://github.com/Significant-Gravitas/AutoGPT
[17] N. Akhtar, S. Rabbani, H. Rabbani, Saurav Kumar, Y. Perwej, "AI-Driven Intelligent Resume Recommendation Engine", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Print ISSN:

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52016

2395-1990, Online ISSN: 2394-4099, Volume 12, No. 3, Pages 1141–1155, June 2025, DOI: 10.32628/IJSRSET2512145.

[18] Prof. Kameswara Rao Poranki, Y. Perwej, Nikhat Akhtar," Integration of SCM and ERP for Competitive Advantage", TIJ's Research Journal of Science & IT Management – RJSITM, International Journal's-Research Journal of Science & IT Management of Singapore, ISSN:2251-1563, Singapore, in www.theinternationaljournal.org as RJSSM, Volume 04, Number 05, Pages 17-24, 2015

[19] Y. Perwej, "An Evaluation of Deep Learning Miniature Concerning in Soft Computing", International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), ISSN (Online): 2278-1021, ISSN (Print): 2319-5940, Volume 4, Issue 2, Pages 10 - 16, February 2015, DOI: 10.17148/IJARCCE.2015.4203

[20] Y. Perwej, Firoj Parwej, "A Neuroplasticity (Brain Plasticity) Approach to Use in Artificial Neural Network", International Journal of Scientific & Engineering Research (IJSER), France, ISSN 2229 – 5518, Volume 3, Issue 6, Pages 1- 9, 2012, DOI: 10.13140/2.1.1693.2808

[21] Y. Perwej, "The Bidirectional Long-Short-Term Memory Neural Network based Word Retrieval for Arabic Documents", Transactions on Machine Learning and Artificial Intelligence (TMLAI), which is published by Society for Science and Education, United Kingdom (UK), ISSN 2054-7390, Volume 3, Issue 1, Pages 16 - 27, 2015, DOI: 10.14738/tmlai.31.863

[22] Y. Perwej, "Unsupervised Feature Learning for Text Pattern Analysis with Emotional Data Collection: A Novel System for Big Data Analytics", IEEE International Conference on Advanced computing Technologies & Applications (ICACTA'22), SCOPUS, IEEE No: #54488 ISBN No Xplore: 978-1-6654-9515-8, Coimbatore, India, 2022, DOI: 10.1109/ICACTA54488.2022.9753501

[23] M. Wooldridge and N. R. Jennings, "Intelligent Agents: Theory and Practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115-152, 1995.

[24] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv:1503.02531*, 2015.

[25] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in NeurIPS*, vol. 33, 2020.

[26] L. Ouyang et al., "Training Language Models to Follow Instructions with Human Feedback," in *Advances in NeurIPS*, vol. 35, 2022.

[27] K. Cobbe et al., "Training Verifiers to Solve Math Word Problems," *arXiv:2110.14168*, 2021.

[28] R. Nakano et al., "WebGPT: Browser-Assisted Question-Answering with Human Feedback," *arXiv:2112.09332*, 2021.

[29] C. Guo et al., "On Calibration of Modern Neural Networks," in *Proc. ICML*, 2017, pp. 1321-1330.

[30] S. Bubeck et al., "Sparks of Artificial General Intelligence: Early Experiments with GPT-4," *arXiv:2303.12528*, 2023.

[31] A. Madaan et al., "Self-Refine: Iterative Refinement with Self-Feedback," in *Advances in NeurIPS*, 2023.